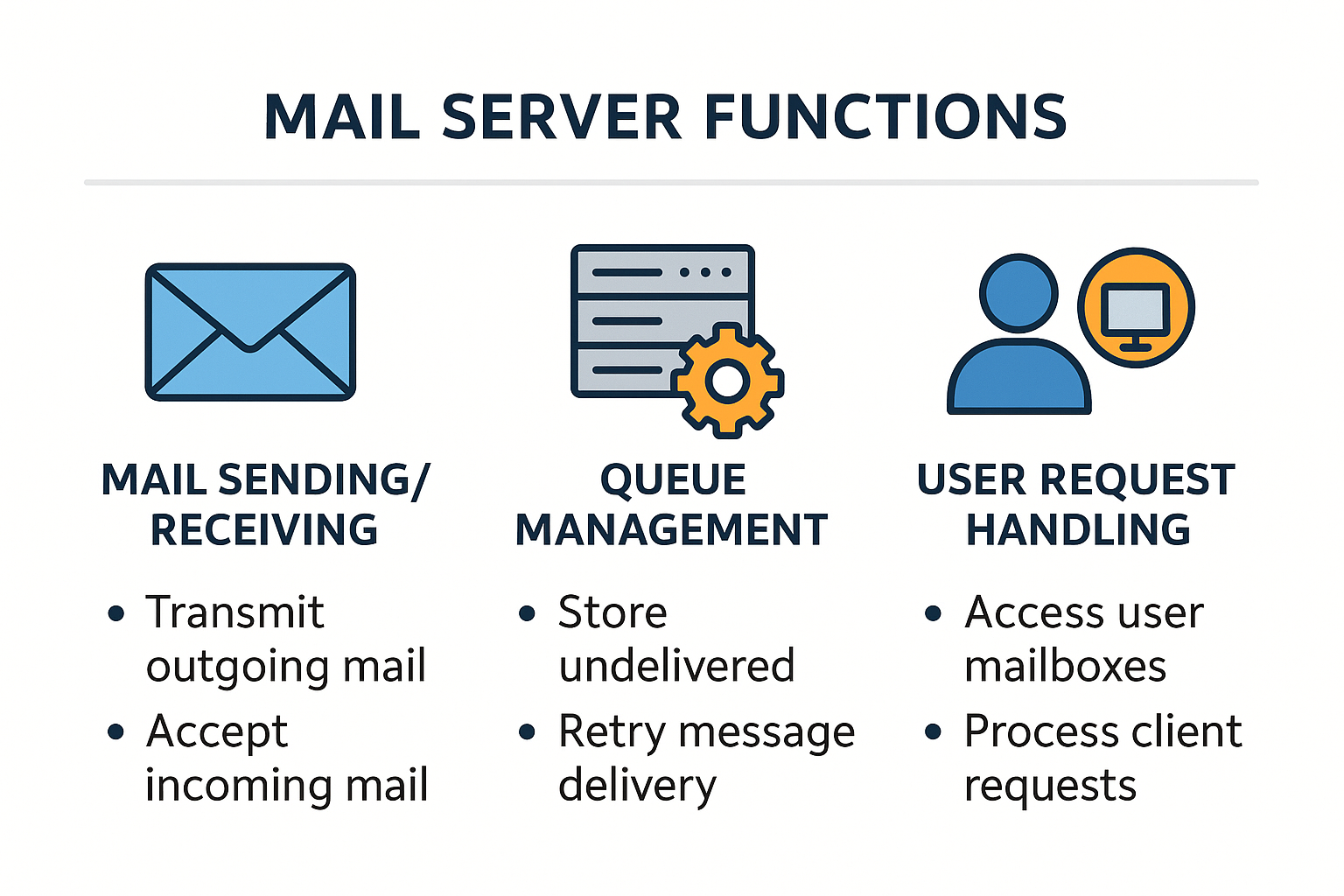
**컴퓨터 네트워크와 클라우드 컴퓨팅**

**소켓 프로그래밍 만들기**

**-서버 ver.**

**<메일 서버의 기능>**



**-메일 송수신이 가능해야 한다.**

**-큐 관리(재수신, 미발신 확인)가 가능해야 한다.**

**-사용자 요청 처리가 가능해야 한다.**

**<코드>**

**import socket**

**import threading**

**import json**

**import time**

**from collections import deque**

**from uuid import uuid4**

**# ====== 서버 설정 ======**

**SERVER\_HOST = 'localhost'**

**SERVER\_PORT = 9001**

**MAX\_CONN = 5**

**BUFFER\_SIZE = 4096**

**# ====== 데이터 저장소 ======**

**class MailSystem:**

**def \_\_init\_\_(self):**

**self.users = {'alice': 'pass1', 'bob': 'pass2'}**

**self.inboxes = {user: deque(maxlen=100) for user in self.users} # 최대 100개 메일 저장**

**self.outbox = deque()**

**self.active\_connections = {} # 실시간 연결된 사용자**

**# ====== 큐 관리 ======**

**def process\_outbox(self):**

**while self.outbox:**

**mail = self.outbox.popleft()**

**receiver = mail['receiver']**

**# 실시간 전송 시도**

**if receiver in self.active\_connections:**

**try:**

**conn = self.active\_connections[receiver]**

**conn.send(json.dumps({**

**'type': 'mail\_push',**

**'mail': mail**

**}).encode())**

**print(f"[큐 관리] 실시간 전송 성공: {receiver}")**

**except Exception as e:**

**print(f"[큐 관리] 실시간 전송 실패: {e}")**

**self.\_store\_to\_inbox(receiver, mail)**

**else:**

**self.\_store\_to\_inbox(receiver, mail)**

**def \_store\_to\_inbox(self, receiver, mail):**

**if receiver in self.inboxes:**

**self.inboxes[receiver].append(mail)**

**print(f"[큐 관리] 인박스 저장: {receiver}")**

**else:**

**print(f"[큐 관리] 잘못된 수신자: {receiver}")**

**# ====== 사용자 요청 처리 ======**

**def handle\_request(self, conn, data):**

**try:**

**req = json.loads(data.decode())**

**handler = getattr(self, f"handle\_{req['type']}", None)**

**return handler(conn, req) if handler else {'status': 'invalid\_request'}**

**except json.JSONDecodeError:**

**return {'status': 'decode\_error'}**

**def handle\_login(self, conn, req):**

**if req['id'] not in self.users:**

**return {'status': 'id\_not\_found'}**

**if self.users[req['id']] != req['pw']:**

**return {'status': 'wrong\_password'}**

**self.active\_connections[req['id']] = conn**

**return {'status': 'success', 'user': req['id']}**

**def handle\_send\_mail(self, \_, req):**

**mail\_id = str(uuid4())**

**new\_mail = {**

**'id': mail\_id,**

**'sender': req['sender'],**

**'receiver': req['receiver'],**

**'subject': req['subject'],**

**'body': req['body'],**

**'timestamp': time.strftime("%Y-%m-%d %H:%M:%S")**

**}**

**self.outbox.append(new\_mail)**

**return {'status': 'queued', 'mail\_id': mail\_id}**

**def handle\_get\_mail\_list(self, \_, req):**

**user = req['user']**

**return {**

**'status': 'success',**

**'mails': [{**

**'id': mail['id'],**

**'subject': mail['subject'],**

**'sender': mail['sender'],**

**'timestamp': mail['timestamp']**

**} for mail in self.inboxes[user]]**

**}**

**def handle\_read\_mail(self, \_, req):**

**user = req['user']**

**target\_id = req['mail\_id']**

**for mail in self.inboxes[user]:**

**if mail['id'] == target\_id:**

**return {'status': 'success', 'mail': mail}**

**return {'status': 'mail\_not\_found'}**

**# ====== 서버 실행 ======**

**class MailServer:**

**def \_\_init\_\_(self):**

**self.system = MailSystem()**

**self.server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**self.server\_socket.bind((SERVER\_HOST, SERVER\_PORT))**

**self.server\_socket.listen(MAX\_CONN)**

**# 큐 처리 스레드 시작**

**threading.Thread(target=self.\_process\_queue, daemon=True).start()**

**print(f"🚀 메일 서버 시작: {SERVER\_HOST}:{SERVER\_PORT}")**

**def \_process\_queue(self):**

**while True:**

**self.system.process\_outbox()**

**time.sleep(1) # 1초마다 큐 처리**

**def run(self):**

**while True:**

**conn, addr = self.server\_socket.accept()**

**threading.Thread(target=self.\_client\_handler, args=(conn,)).start()**

**def \_client\_handler(self, conn):**

**try:**

**while True:**

**data = conn.recv(BUFFER\_SIZE)**

**if not data: break**

**response = self.system.handle\_request(conn, data)**

**conn.send(json.dumps(response).encode())**

**finally:**

**conn.close()**

**# ====== 클라이언트 ======**

**class MailClient:**

**def \_\_init\_\_(self):**

**self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)**

**self.sock.connect((SERVER\_HOST, SERVER\_PORT))**

**self.user = None**

**# 실시간 알림 수신 스레드**

**threading.Thread(target=self.\_receive\_notifications, daemon=True).start()**

**def \_receive\_notifications(self):**

**while True:**

**try:**

**data = self.sock.recv(BUFFER\_SIZE)**

**if data:**

**notification = json.loads(data.decode())**

**if notification['type'] == 'mail\_push':**

**print(f"\n🔔 새 메일: {notification['mail']['subject']}")**

**except:**

**break**

**def login(self, user\_id, password):**

**req = {'type': 'login', 'id': user\_id, 'pw': password}**

**self.sock.send(json.dumps(req).encode())**

**res = json.loads(self.sock.recv(BUFFER\_SIZE).decode())**

**if res['status'] == 'success':**

**self.user = user\_id**

**return res**

**def send\_mail(self, receiver, subject, body):**

**req = {**

**'type': 'send\_mail',**

**'sender': self.user,**

**'receiver': receiver,**

**'subject': subject,**

**'body': body**

**}**

**self.sock.send(json.dumps(req).encode())**

**return json.loads(self.sock.recv(BUFFER\_SIZE).decode())**

**def get\_mail\_list(self):**

**req = {'type': 'get\_mail\_list', 'user': self.user}**

**self.sock.send(json.dumps(req).encode())**

**return json.loads(self.sock.recv(BUFFER\_SIZE).decode())**

**def read\_mail(self, mail\_id):**

**req = {'type': 'read\_mail', 'user': self.user, 'mail\_id': mail\_id}**

**self.sock.send(json.dumps(req).encode())**

**return json.loads(self.sock.recv(BUFFER\_SIZE).decode())**

**# ====== 실행 예시 ======**

**if \_\_name\_\_ == "\_\_main\_\_":**

**import sys**

**if '--server' in sys.argv:**

**MailServer().run()**

**else:**

**client = MailClient()**

**print("=== 메일 클라이언트 ===")**

**user = input("사용자 ID: ")**

**pw = input("비밀번호: ")**

**res = client.login(user, pw)**

**if res['status'] != 'success':**

**print("❌ 로그인 실패")**

**exit()**

**print("명령어: send, list, read, exit")**

**while True:**

**cmd = input("> ").strip().lower()**

**if cmd == 'send':**

**to = input("수신자: ")**

**subject = input("제목: ")**

**body = input("내용: ")**

**print(client.send\_mail(to, subject, body))**

**elif cmd == 'list':**

**mails = client.get\_mail\_list()**

**for idx, mail in enumerate(mails.get('mails', [])):**

**print(f"{idx+1}. {mail['subject']} ({mail['sender']})")**

**elif cmd == 'read':**

**mail\_id = input("메일 ID: ")**

**print(client.read\_mail(mail\_id))**

**elif cmd == 'exit':**

**break**